A Hybrid Approach to Exascale Operating Systems

Yoonho Park* Eric Van Hensbergen Marius Hillenbrand Todd Inglett Bryan Rosenburg Kyung Dong Ryu
IBM Research and Systems Technology Group

I. Challenges Addressed

In order to reach exascale, systems will require several orders of magnitude improvement in node-level performance and power efficiency. This aggressive power/performance challenge has led hardware vendors to re-evaluate node-level processor architectures. While it is anticipated that processors will continue to follow the general trend towards massive multi-core to increase performance, there is an increased focus on how the cores themselves may be simplified to optimize for power efficiency and density. Additionally, node-level components traditionally considered to be peripherals such as networking interfaces, secondary storage, and even memory are being imbued with their own processing elements and purpose-built accelerators. The programmable nature of this heterogeneous set of system elements broadens the scope of responsibility for systems software, transforming the node-level operating system and runtime into a distributed system in its own right.

At the same time, there is an increased call for high performance computing system software to provide richer (and more familiar to the desktop) environments allowing a broader range of applications. This has implications across the full range of system software. It results in a need to support the capabilities provided by a traditional, time-shared, general-purpose operating system, such as Linux, including libraries, file systems, and daemon-provided services. One challenge with just using Linux is that historically, Linux developers have been reticent to adopt changes specific to the HPC community. This is in part because the Linux community tends to accept changes that matter for the general population, while HPC architectures have tended to push technology limits in order to achieve the highest performance for scientific and engineering applications.

Traditionally, HPC operating systems have fallen into one of two categories: full-weight kernels (FWK) such as Linux and light-weight kernels (LWK) such as Blue Gene's Compute Node Kernel (CNK) [5] or Sandia's Catamount kernel for Cray systems [7]. Instead of forcing users and applications into one of these two extremes, we propose a hybrid approach which allows the co-existence of both environments on multi-core platforms while at the same time extending support to emerging heterogeneous computing environments. We call this approach a *FusedOS* [10].

II. Maturity

In order to study these issues, we implemented a prototype FusedOS infrastructure on Blue Gene/Q. Although Blue Gene/Q has homogeneous cores, we simulate heterogeneous cores by dedicating a set of cores to applications and running Linux only on the 17th core as shown in Figure 1. In that role, cores run almost exclusively in user-mode executing application code. A small supervisor-state monitor is used only to simulate the hardware we would expect to help manage such a heterogeneous environment and forward system calls and exceptions to the cores running the full-weight kernel.

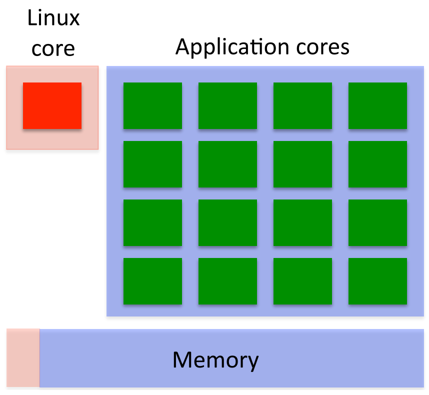

Fig. 1. Partitioning of cores and memory between Linux and applications.

For our prototype we focused on execution of existing Blue Gene/Q applications. CNK supervisor code is packaged as a library we call CL and runs as part of a shadow user process on Linux. It is responsible for launching the workload on the application cores and servicing exceptions and requests while the application is running. Because it is running as a user process, it can be managed in familiar ways and has a normal user context for accessing Linux resources such as the file system, network, etc.

Our current benchmark results indicate that there are performance and noise advantages to our approach even on a homogeneous system. Extensive evaluation comparing OS noise using the FTQ benchmark between FusedOS, Linux, and CNK have been made showing FusedOS noise characteristics equivalent to CNK which essentially demonstrates no noise at all. Additionally, these same evaluations showed dangerous noise trends in Linux on massive multicore systems both with and without the use of hardware threading mechanisms. Performance between FusedOS and CNK was also roughly equivalent, with some surprising cases where FusedOS outperformed CNK and others where the remote system call overhead caused a slight degradation of performance for FusedOS. In all cases, the application on FusedOS outperformed Linux.

III. Uniqueness

The challenges of multicore and heterogenous systems are not necessarily unique to exascale computing, however high

*Contact author: yoonho@us.ibm.com

performance computing applications tend to have tighter constraints on aspects such as OS interference, power efficiency, and compute performance than typical commerical applications. Additionally, the aggressive power/performance targets of exascale will drive more radical approaches to system architectures requiring the use of such an approach sooner for exascale than it would be required for commerical computing.

IV. Novelty

It may be tempting to compare our approach to microkernel or more specifically exokernel [4] efforts of the past. While aspects of our approach are closely derived from such efforts, we believe the hybrid kernel approach in conjunction with the specific requiremenets and nature of high performance computing applications to be a different enough design point to justify further exploration and investigation.

Similarly, it is valid to ask why we did not use virtualization as a mechanism for partitioning node resources and deploying applications directly on top of the hypervisor layer. An approach similar to this was taken by some of the co-authors in the past for both high-performance [6] and commercial [2] workloads. While this approach showed promise, there was a higher degree of overhead for forwarding operations to the full-weight kernel due to the additional protection mechanisms of virtualization. While modern general-purpose processors are making great strides at reducing these overheads, a founding premise of the FusedOS project is that aggressive power-efficiency and density targets for exascale processors would prohibit more complex features such as hardware accelerated virtualization within the processor.

Another approach, taken by ZeptoOS [1], is to start with a Linux image and make modifications such as reserving memory during boot before Linux accesses it, and removing auxiliary daemons to reduce the noise. By doing so, ZeptoOS can study operating system issues for petascale architectures with ten thousand to one million CPUs. A similar study undertaken by Shmueli et al. [11] looked at the issues that caused Linux not to scale to tens of thousands of nodes. These groups attempt to maintain scalability by reducing daemons and working on memory allocation issues directly within Linux on homogeneous systems. We believe our approach is complementary to such an environment in that we also address heterogeneous systems and provide more architected mechanisms for dynamic resource reservation and assignment to applications or system services.

NIX [9] is an effort closely related to our work. Motivated by personal communication between the teams, its focus is primarily on role assignment of cores. They are exploring three different types of roles: time-shared cores (TCs) that run in a manner similar to a general-purpose operating system, kernel cores (KCs) that run only in supervisor mode and handle system services and device drivers as well as system service requests from applications, application cores (ACs) that run applications and forward all system calls to kernel-mode processes in TCs or KCs. While NIX has proven to be a great testbed for evaluating systems concepts, its user environment is different enough to make running existing HPC applications very difficult. By contrast our FusedOS approach can run existing CNK applications without modification, and could easily be extended to run vanilla Linux binaries within the application cores.

V. Applicability

Our early prototype illustrates opportunities for co-design of heterogeneous processor architectures, heterogeneous nodes, and heterogeneous platforms as a whole. Exascale architectures may move away from heterogeneous environments, but we have found performance benefits even within homogeneous environments. Beyond dedicating processing cores to applications, we believe there is also opportunity to look into dedicating cores to device drivers or other kernel services in order to isolate critical services or sensitive intellectual property. Furthermore, by taking advantage of isolated cores we can also explore deploying more traditional exokernel library OS mechanisms or other experimental kernels without impacting the rest of the system. While we have prototyped the infrastructure on Blue Gene/Q, the design focus on targeting heterogeneity naturally allows it to be applied easily to different architectures and machine topologies.

VI. Effort

There are a number of important design trade-offs to be studied from an architectural co-design perspective as well as a number of core operating system design principles which need to be re-evaluated in such an environment. Additionally, our hybrid kernel approach needs to be more closely compared to contemporary alternative infrastructures incorporating processor virtualization such as Palacios [8] and multi-kernel environments such as Barrelfish [3].

Defining the right API for coordinating deployment of code across node-local heterogeneous cores, peripherals, and the broader cluster platform is something that will require iterative effort and co-design with various run-time and programming model teams. Elements of the API may motivate changes in the underlying architecture, and may also require new interfaces in the vendor's system management tools.

While a modest amount of effort could provide simple extensions of our existing prototype helping further validate the idea, a broader agenda would allow us and interested partners to more fully explore the design space, evaluate a larger set of scientific applications at scale, tailor environments to specific classes of applications, more tightly integrate with runtimes, and explore incorporating both isolation for reliability as well as feedback-directed self-aware dynamic policy management of node- and platform-level partitioning and workflow component deployment.

Acknowledgements

We would like to thank the following people for providing the Linux kernel and build environment used with FusedOS: Blake Fitch (IBM Watson Blue Gene Active Storage group), Heiko Schick, Peter Morjan (IBM Boebligen Exascale Innovation Center), and Thomas Gooding (IBM Rochester Blue Gene kernel team). The motivations behind the hybrid approach concept came out of conversations with Robert Wisniewski while he was still at IBM Research.

The Blue Gene/Q project has been supported and partially funded by Argonne National Laboratory and the Lawrence Livermore National Laboratory on behalf of the U.S. Department of Energy, under Lawrence Livermore National Laboratory subcontract no. B554331. Aspects of the early evaluation of the FusedOS prototype was also supported by the U.S. Department of Energy under Award Numbers DE-SC0005365 and DE-SC0007103.

References

[1] ZeptoOS: The small Linux for big computers. http://www.mcs.anl.gov/research/projects/zeptoos/.

[2] G. Ammons, J. Appavoo, M. Butrico, D. Da Silva, D. Grove, K. Kawachiya, O. Krieger, B. Rosenburg, E. Van Hensbergen, and R. W. Wisniewski. Libra: A library operating system for a JVM in a virtualized execution environment. In *3rd International Conference on Virtual Execution Environments*, VEE '07, pages 44–54, New York, NY, USA, 2007. ACM.

[3] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The Multikernel: A new OS architecture for scalable multicore systems. In *ACM Symposium on Operating Systems Principles*, pages 29–44, New York, NY, USA, 2009. ACM.

[4] D. R. Engler, M. F. Kaashoek, and J. O'Toole Jr. Exokernel: An operating system architecture for application-level resource management. In *ACM Symposium on Operating System Principles*, pages 251–266, 3–6 December 1995.

[5] M. Giampapa, T. Gooding, T. Inglett, and R. W. Wisniewski. Experiences with a lightweight supercomputer kernel: Lessons learned from Blue Gene's CNK. In *ACM/IEEE International Conference for High Performance Computing (SC10)*, New Orleans, LA, November 2010.

[6] E. V. Hensbergen. PROSE: partitioned reliable operating system environment. *ACM SIGOPS Operating Systems Review*, 40(2):12–15, 2006.

[7] S. Kelly and R. Brightwell. Software architecture of the lightweight kernel, Catamount. In *Cray Users' Group Annual Technical Conference*, Albuquerque, New Mexico, June 2005.

[8] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, and R. Brightwell. Palacios: A new open source virtual machine monitor for scalable high performance computing. In *IEEE International Parallel & Distributed Processing Symposium*, April 2010.

[9] R. Minnich, N. Evans, E. Soriano, F. Ballesteros, J. McKie, G. Guardiola, and C. Forsyth. High performance cloud computing is NIX. *Bell Labs Technical Conference*, 2011.

[10] Y. Park, E. V. Hensbergen, M. Hillenbrand, T. Inglett, B. Rosenburg, K. Ryu, and R. Wisniewski. FusedOS: Fusing LWK performance with FWK functionality in a heterogeneous environment. In *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2012. (Draft under submission).

[11] E. Shmueli, G. Almási, J. Brunheroto, J. Castaños, G. Dózsa, S. Kumar, and D. Lieber. Evaluating the effect of replacing CNK with Linux on the compute-nodes of Blue Gene/L. In *22nd ACM International Conference on Supercomputing (ICS 2008)*, Island of Kos, Greece, June 2008.